E.T.S. de Ingeniería Industrial,
Informática y de Telecomunicación

Detección de actividad sospechosa en la red asociada con ataques cibernéticos en el ámbito ofimático

Grado en Ingeniería
en Tecnologías de Telecomunicación

Trabajo Fin de Grado

Iván José Caballero Bocanegra

Eduardo Magaña Lizarrondo

Pamplona, 12 de mayo de 2020

upna Universidad Pública de Navarra
Nafarroako Unibertsitate Publikoa

Resumen

Se desarrolla un programa de tipo modular con el objetivo de detectar actividad sospechosa en la red en función de los módulos utilizados. Consiste en un programa principal que contiene las funciones necesarias para analizar la actividad de red, como leer las trazas y su contenido, generar una serie de registros, y dar posibles avisos en caso de encontrar patrones de actividad de malware sin requerir intervención del usuario. La actividad por detectar queda definida en los módulos y se basa en ciertos paquetes y peticiones característicos de los ataques de red, así como en patrones y su frecuencia de repetición. Se han definido módulos para detección de cifrado de archivos compartidos por ransomware y escaneos de máquinas y puertos en red.

Se comprueba su funcionamiento en una serie de pruebas experimentales configurando escenarios virtuales y simulando el cifrado de ficheros compartidos, o analizando capturas de paquetes de actividad de cifrado de distintas variedades de ransomware.

Lista palabras clave

Seguridad de red, detección de malware, ransomware, análisis de tráfico, ataque de red, Python.

Abstract

A modular program is developed with the aim of detecting suspicious activity on the network depending on the modules used. It consists of a main program, which after an analysis for efficient operation, contains the necessary functions to examine network activity, such as reading the traces and their contents, generating a series of logs, and giving warnings in case patterns of malware activity are found without requiring any user input. This activity is defined in the modules and is based on certain packets and requests characteristic of network attacks, as well as patterns and their repeat frequency. Modules have been defined for encryption detection of shared files by ransomware and scans of machines and ports.

It is evaluated in a series of experimental tests by setting up virtual scenarios and simulating the encryption of shared files, as well as examining captures of encryption activity of different varieties of ransomware.

Keywords

Network security, malware detection, ransomware, traffic analysis, network attack, Python.

Índice

Introducción

Capítulo 1. Introducción

Entre los mayores problemas y retos para las empresas en cuestiones de seguridad en sus redes internas se encuentra la intrusión y propagación de programas maliciosos o malware por sus redes corporativas. Estas empresas generalmente prestan una especial atención a la seguridad perimetral, manteniéndola altamente preparada ante ataques con el uso de cortafuegos.

Uno de los principales vectores de ataques maliciosos en Europa es el correo electrónico, ya sean correos que contienen malware o enlaces maliciosos. Hacen uso de técnicas de ingeniería social como es el caso de phishing, uno de los métodos más utilizados para introducirse en un equipo de la red corporativa de las empresas. Otra técnica es el uso de ataques zero-day para los cuales las herramientas de protección antimalware no están preparadas y consiguen comprometer las máquinas. Estos ataques tienen como objetivo común obtener un lucro económico, ya sea por suplantación de identidad, robo de credenciales bancarias o datos confidenciales y posterior rescate en criptomonedas, etc [1].

La revolución del Internet de las Cosas y la Industria 4.0 está provocando que los destinatarios de esos ataques no solo sean los equipos informáticos tradicionales (servidores, bases de datos, ordenadores personales, etc.) sino también dispositivos industriales que puedan poner en riesgo la fabricación y la actividad económica de las empresas. De hecho, se ha observado durante los dos últimos años una tendencia por el incremento de ataques especialmente dirigidos a empresas y grandes compañías, centros sanitarios y educativos. Algunos de los más extendidos han sido de los troyanos Emotet y Trickbot, que se ha utilizado como servicio de distribución de otros códigos dañinos como ransomware [2].

El ransomware es un tipo de malware que impide a los usuarios acceder a su sistema, el llamado bloqueador de pantalla, o a sus archivos personales, el ransomware de cifrado; exigiendo el pago de un rescate para poder recuperarlos. Sin embargo, no se cifran solo los ficheros de la máquina infectada. La mayoría de las empresas hacen uso de unidades de red compartidas de forma que múltiples usuarios pueden tener acceso a los documentos alojados en ellas, entre otras ventajas para configuración de perfiles, trabajo en equipo o realización de copias de seguridad. Estas unidades suelen ser accesibles por las máquinas infectadas y pueden verse comprometidas. Este tipo de programas son una de las amenazas más destacadas en cuanto a predominancia y perjuicio económico según el último informe IOCTA de Europol [3]. En caso de detectarse una infección de ransomware, se debe parar la actividad empresarial para eliminar el malware y restaurar copias de seguridad, lo que conlleva pérdidas por paradas en cadenas de producción [4] [5]. Una vez el malware se ha infiltrado en una máquina de la compañía, procede a infectar otras máquinas con un comportamiento de gusano, extendiéndose con un movimiento horizontal por la red interna.

Este proyecto se ha centrado en la detección de malware durante este movimiento lateral y otra actividad en la red interna, una vez una máquina ha sido ya infectada. Más en concreto, se han configurado una serie de módulos para detectar actividad de ransomware en

la red. Estos códigos maliciosos realizan el cifrado de ficheros de unidades compartidas de red, las cuales normalmente utilizan el protocolo Server Message Block (SMB) para su gestión. También hacen una serie de escaneos de la red para descubrir otras máquinas a las que propagarse. En ambos casos, se efectúan varias peticiones de forma masiva que siguen una serie de patrones y se pueden detectar analizando los paquetes en red.

1.1 Contexto tecnológico

Tradicionalmente se ha llamado "antivirus" a aquellos softwares encargados de detectar y eliminar todo tipo de programas maliciosos o malware. No obstante, debido al enorme crecimiento en el número de este tipo de amenazas, estos programas han evolucionado considerablemente, apareciendo en el mercado numerosas soluciones de mayor complejidad. Esto ha sucedido en consecuencia del aumento de la complejidad de estos códigos maliciosos, ya sea a nivel de la propia codificación, del uso de técnicas de ofuscación como aquellos polimórficos o variación de los vectores de distribución. Las técnicas para detección de malware pueden basarse en comportamiento, en heurística o en firmas. Los basados en comportamiento analizan las acciones que realiza el malware durante el proceso de infección para posteriormente detectar estos comportamientos en los equipos, como llamadas de sistema, cambios en ficheros, monitorización de actividad de red o análisis de procesos [6]. Aquellas heurísticas hacen uso de reglas y algoritmos que buscan comandos como llamadas API o relaciones de archivos que puedan indicar intenciones maliciosas [7]. Y por último las basadas en firmas, que generan una firma de un fichero y se busca en una base de datos de firmas de malware ya conocidos [8]. Los softwares antimalware combinan las distintas técnicas para conseguir una mayor precisión.

Entre estos programas para la detección de programas maliciosos se encuentra los Sistemas de Detección de Intrusos o IDS. Como su nombre dice, tienen como función principal identificar o detectar la presencia de actividades intrusivas y emitir una alerta. Por norma general, estas soluciones no cuentan con opciones para detener los ataques por sí solos, sino que requieren de otras herramientas complementarias para realizar esta tarea, como firewalls para efectuar un bloqueo.

Los IDS se pueden clasificar en dos tipos: Host IDS y Network IDS. Los HIDS monitorizan un equipo específico, analizando el tráfico entrante y saliente, procesos a nivel de sistema operativo, entre otras actividades a nivel de host. Deben adaptarse a la plataforma que monitorizan, ya que los tipos de ficheros y registros varían entre sistemas operativos. Tienen una mayor lentitud de respuesta en comparación a los NIDS dado que se limitan a analizar los registros de actividad, y descubren los ataques en un estado avanzado o cuando ya han tenido lugar. Por lo que se refiere a los NIDS, se conectan a la red y observan el tráfico de paquetes en busca de anomalías que puedan indicar una intrusión. Comprueban cada paquete para

determinar si corresponde con algún tipo de ataque conocido. Pueden detectar ataques en todos los equipos conectados a un mismo segmento de la red, independientemente de la plataforma utilizada por los equipos, resultando así invisibles para el equipo atacante. Sin embargo, el análisis con esta herramienta en redes de alta velocidad o en aquellas en las que el tráfico está cifrado no es viable. No obstante, ambos tipos pueden usarse en conjunto para compensar algunas de las desventajas [9].

Una evolución de estos sistemas anteriores son los Sistemas de Prevención de Intrusos o IPS. Analizan la red y realizan acciones automáticas en los flujos de tráfico como descartar tráfico malicioso, bloquear el tráfico proveniente de cierta dirección IP o realizar un reset de la conexión. Para esto, todo el tráfico se enruta a través de ellos, de manera que debe hacer un procesado rápido y eficiente [10]. Un ejemplo de este tipo de sistemas es Snort [11], de código abierto y capaz de realizar un análisis de protocolos y búsqueda de contenido para detectar ataques de buffer overflow, escaneos de puertos, fingerprinting del sistema operativo, etc.

1.2 Prácticas curriculares en empresa

Este Trabajo de Fin de Grado se ha iniciado durante la estancia de prácticas curriculares en BSH Electrodomésticos España S.A, en el Centro de Servicios Corporativos de Huarte.

En un principio, el periodo de prácticas estaba planteado desde el 10 de febrero al 19 de junio, pero debido al estado de alarma declarado por el COVID-19 continuaron en forma de teletrabajo en un principio, pero finalmente se suspendieron temporalmente desde el 1 de abril. Por ello, el proyecto se ha continuado aunque las distintas pruebas que se han realizado han sido con las consiguientes limitaciones de recursos y escenario disponibles, configurando escenarios de red virtuales en los casos posibles.

Capítulo 2. Objetivos

Uno de los principales problemas y retos de las empresas en cuestiones de seguridad en sus redes internas es la intrusión y propagación de programas maliciosos o malware. Estas empresas generalmente prestan una especial atención a la seguridad perimetral, manteniéndola altamente preparada ante ataques con el uso de cortafuegos, sin prestar tanta atención a la actividad interna.

Se plantea el desarrollo de un programa de tipo modular con el objetivo de detectar actividad sospechosa en la red en función de los módulos utilizados. Consistiría en un programa principal o Core, que previo análisis para un funcionamiento eficiente, contendría todas las funciones comunes necesarias para analizar la actividad de red, como leer las trazas y su contenido, generar una serie de registros, y dar posibles avisos en caso de encontrar patrones de actividad de malware sin requerir intervención del usuario. Debe poder ejecutarse tanto en máquinas Windows como Linux, principales sistemas operativos en entornos corporativos. Se configurarían módulos para detección de ransomware, en concreto para el cifrado de ficheros en directorios compartidos en red y escaneos de hosts.

Se comprobará su funcionamiento en una serie de pruebas experimentales configurando escenarios virtuales y simulando el cifrado de ficheros compartidos, así como analizando capturas de paquetes de actividad de ransomware.

Capítulo 3. Metodología

3.1 Herramientas empleadas

Este proyecto se ha desarrollado y probado haciendo uso de los siguientes equipos:

- Ordenador Fujitsu Lifebook E746: Intel® Core i5-6 Gen CPU, 8 GB RAM, tarjetas de red Intel® I219 Gigabit, Windows 10.
- Ordenador Xiaomi Notebook Pro 1Gen: Intel® Core i5-8 Gen CPU, 8 GB RAM, tarjetas de red Realtek® Gigabit, Windows 10.

El programa se ha desarrollado en Python en su versión 3.8. Es un lenguaje de programación interpretado, orientado a objetos y de código abierto. Tiene una sintaxis simple, pero permite programar a alto nivel y aplicarse en múltiples aplicaciones [12].

Se ha elegido por su gran polivalencia y por su uso ampliamente extendido. Está presente en ámbitos variados, desde el educacional para la enseñanza en escuelas y centros de formación, hasta el industrial para control de calidad o producción lean (*Lean Manufacturing*) [12]. Personalmente, ha sido una buena oportunidad para aprender este lenguaje. Además, cuenta con una amplia comunidad que ha desarrollado proyectos y librerías para diversos casos concretos, que se pueden encontrar repositorios de código públicos como Python Packaging Index o GitHub.

El entorno de desarrollo elegido es PyCharm en su versión gratuita *Community Edition* 2020.1. Cuenta con opciones de inspección de código, depurador, navegación y refactorización. Asimismo, ha resultado de gran ayuda por las numerosas recomendaciones y buenas prácticas que ofrece a la hora de trabajar con este lenguaje.

Las librerías externas de Python empleadas en el proyecto y es necesario instalar han sido las siguientes:

- **Scapy** [13]: es un programa y librería de Python que permite manipular de forma interactiva paquetes de red. Tiene opciones de captura, decodificación y análisis entre muchos otros.
- **Regex** [14]: es una librería utilizada para buscar patrones o expresiones regulares en cadenas de caracteres, entre otras funciones.
- **SQLAlchemy** [15]: es una librería que engloba una serie de herramientas para manejar una base de datos SQL y proporciona un mapeo objeto-relacional (*Object Relational Mapper*) entre objetos de Python y de la base de datos.

3.2 Otras herramientas

Como herramienta para capturar paquetes de red y realizar las comprobaciones requeridas comparando resultados, como paquetes analizados o los que cumplen ciertas condiciones, se ha usado Wireshark en la versión 3.2.2.

En la fase experimental y pruebas de este proyecto se ha empleado Virtual Box en la versión 6.0.14 como herramienta para ejecutar una serie de máquinas virtuales a modo de atacante (máquina infectada), víctima y máquina para detección. Las pruebas se han llevado a cabo virtualizando máquinas con sistema operativo Windows 10 y Kali Linux versión 2020.01. La conexión entre estas máquinas es de tipo *Host only*, es decir, se crea una red virtual aislada entre las máquinas virtuales y el equipo anfitrión. Se comentará con mayor detalle en el capítulo 5.

3.3 Caracterización del comportamiento del ransomware

Previamente al desarrollo del programa y sus módulos, se han analizado las necesidades y posibles requerimientos con los que debía cumplir.

Una vez una máquina ha sido infectada, el ransomware encriptará el máximo número de documentos posible, utilizando varios hilos de ejecución y dando prioridad a documentos de cierto tamaño para causar el mayor impacto posible.

En escenarios con un servidor de ficheros compartidos, estos ficheros están disponibles desde las distintas máquinas cliente. En estos escenarios, por ejemplo, un equipo en una oficina puede acceder a varios servidores de ficheros compartidos de diferentes directorios, con los documentos y proyectos de distintos equipos o departamentos. El ransomware encriptará estos ficheros de las unidades de red provocando una parada en el flujo de trabajo, lo que supone un claro perjuicio a la actividad de la empresa. Si la empresa no dispone de copias de seguridad o estas han sido encriptadas (ha conseguido elevar privilegios a través de un exploit y acceder a un directorio que requería permisos) tendrá una mayor presión en pagar el rescate para evitar la pérdida total.

La gestión de estos directorios en red se lleva a cabo por el protocolo Server Message Block (SMB). Se realizan una serie de peticiones a la hora de acceder a los ficheros. Generalmente se genera una petición del fichero llamada Create, y le sigue la acción que se requiera Read/Write ya sea de lectura o escritura del fichero respectivamente. Estas peticiones se pretenden analizar, tanto en volumen como en ratio para la detección.

3.4 Prueba de librerías

Se ha elegido la librería de Python Scapy para capturar e implementar una serie de análisis de los paquetes de red. Se barajaron otras opciones como Pyshark o Pypcap pero finalmente quedaron descartadas por ser menos versátiles y completas.

Se realizaron una serie de pruebas con la librería implementando un sniffer de paquetes sencillo y aplicando filtros. Tiene implementados numerosos protocolos de las diferentes capas de la torre OSI como de nivel de enlace (Ethernet, PPP, etc.), nivel de red (IP, ARP, ICMP, etc.), así como de transporte (TCP, UDP, PPTP, etc.), entre muchos otros.

En una primera prueba se intenta localizar paquetes dirigidos a cierta dirección IP, la del servidor de ficheros compartidos, y al puerto TCP destino 445, el que emplea el protocolo SMB. El script localiza los paquetes sin problema. Se prueba a analizar los paquetes de red directamente y también desde un fichero PCAP.

Posteriormente, a partir del script anterior, se añade un análisis por expresiones regulares de la carga de los paquetes. Con la librería *Regex* y aplicando el formato necesario se configura una serie de expresiones para coincidir con la cabecera y ciertos parámetros del protocolo SMB. Para ello, ha sido necesario el estudio en detalle del protocolo en la versión 2. Esta librería permite buscar cierta expresión en una cadena de caracteres o *string*, en este caso es la carga de nivel de aplicación de los paquetes en formato hexadecimal. Tras una serie de pruebas realizadas, se consigue una mayor rapidez en el análisis empleando variables de tipo objetos de bytes en lugar de cadenas de caracteres. Se localizan distintos paquetes y peticiones del protocolo satisfactoriamente, entre otros parámetros de los paquetes de red.

Por otro lado, se han realizado otros scripts de prueba de librerías para verificar y validar su funcionamiento. Se configura una base de datos empleando el ORM SQLAlchemy, configurando unas columnas de prueba para realizar un testeo de las distintas consultas de inserción, obtención y borrado de datos. También se prueban las consultas con filtros que serían necesarias en el programa. Por último, con el fin de incluir un fichero de configuración para que el programa obtenga los distintos parámetros desde este, se probó la librería *configparser*.

Capítulo 4. Arquitectura del sistema

4.1 Descripción general del proyecto

Este proyecto se ha desarrollado con la idea de una herramienta personalizable y multiplataforma. Se trata de una solución modular y se plantea el despliegue en una máquina en un punto intermedio de la red.

Numerosas empresas disponen de redes muy específicas con topologías propias ya sean redes ofimáticas o las redes OT (*Operational Technologies*). Estas últimas tienen sistemas SCADA y PLCs muy heterogéneas que se suelen caracterizar por carecer de medidas de seguridad. Es por esta razón de su carácter modular, de forma que sea configurable para distintos tipos de amenazas, en base a patrones en paquetes o repeticiones, según las necesidades en el escenario. Se podría establecer para un escenario de una red corporativa con distintas oficinas y departamentos de una empresa con módulos específicos para este caso, o en una red OT con módulos para detectar amenazas en este tipo de redes.

Aunque los servidores estén protegidos por un software antivirus, en muchos casos no llegan a estar infectados, sino que es uno o varios de los clientes de la red los que están comprometidos. Haciendo uso de permisos de administrador atacan de forma remota al servidor. Revisar las modificaciones de ficheros en cada uno de los equipos resulta más costoso por el número de licencias requeridas y puede afectar al rendimiento de los equipos [16]. Soluciones para equipos individuales conllevan un mayor trabajo para la instalación y el mantenimiento diario, además de que pueden suponer pérdidas de tiempo si presentan problemas con los distintos equipos corporativos de los usuarios. También es necesario tener en cuenta el tráfico desde y hacia redes externas.

Se plantea un programa que analice el tráfico de la red y si encuentra indicios de esta actividad sospechosa, lance un aviso a los administradores IT. Por ello, se plantea un esquema de uso en un punto intermedio de red, siguiendo el esquema de la figura 1. Por ejemplo, en un escenario para detectar actividad de cifrado por ransomware, el conmutador se encuentra entre el servidor de ficheros compartidos y los equipos de los usuarios. Este conmutador hace port mirroring redirigiendo el tráfico que lo atraviesa por el puerto donde está conectado el equipo sonda.

Arquitectura del sistema

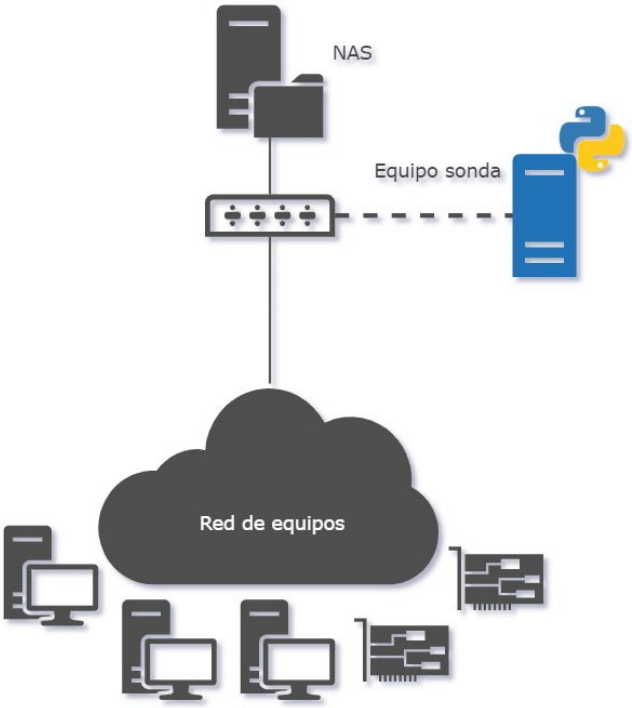


Figura 1. *Escenario de red de ejemplo con equipo sonda conectado al conmutador.*

Este proyecto está formado principalmente por dos partes: el programa principal o Core y los módulos complementarios que el programa carga para efectuar el análisis, como se muestra en el esquema de la figura 2. También cuenta con un fichero con los parámetros de configuración y el de registros de log. Se emplea una base de datos sqlite donde se almacenan registros de paquetes analizados generados por el programa.

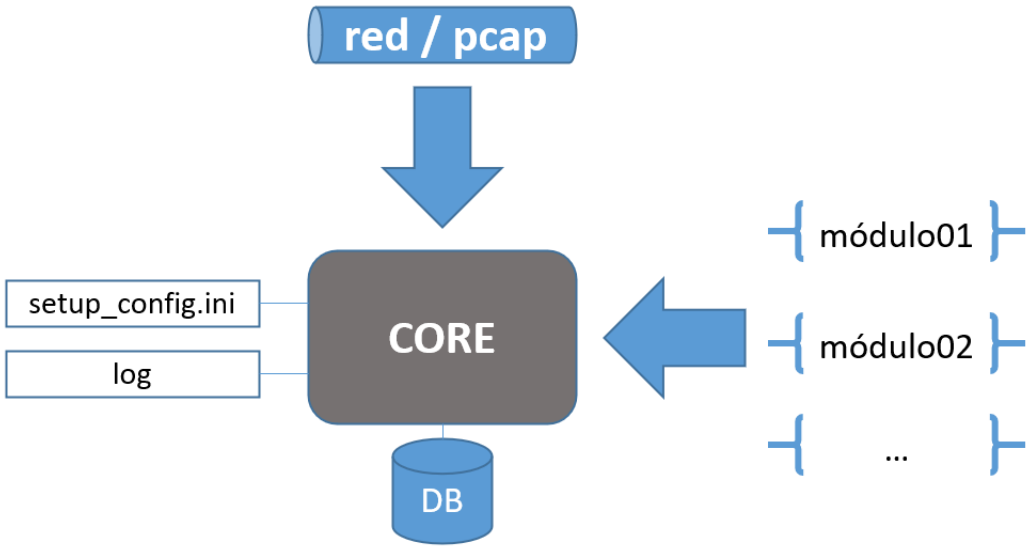


Figura 2. Esquema general del programa, fichero de configuración, fichero de logs, base de datos y módulos.

4.2 Descripción del programa principal

El programa desarrollado realiza análisis de los paquetes de red, buscando paquetes definidos por los módulos. Trabaja de forma pasiva a modo de sniffer de paquetes sin generar tráfico en ningún momento. El análisis lo realiza de forma estándar, evaluando las distintas condiciones y propiedades definidas en los módulos para todos los paquetes.

Permite tanto leer paquetes desde un fichero PCAP como de un interfaz de red en vivo de la máquina en función con los parámetros de configuración que se hayan establecido.

Al tratarse de una herramienta modular, el algoritmo debe estar preparado para detectar todo tipo de paquetes en base a unos parámetros estandarizados en los módulos. Esto incluye los diversos protocolos y sus campos de los siete niveles del modelo OSI. Es por esto por lo que se adapta a los numerosos protocolos definidos en la librería Scapy ya comentada previamente. De manera adicional, y dado que esta librería no tiene definidos todos los protocolos, permite una posterior búsqueda por expresiones regulares. Esto último abre el campo de detección con innumerables posibilidades, si bien también sea dicho, resulta de mayor complejidad establecer estas expresiones, ya que requiere un estudio previo del protocolo de los paquetes a detectar.

El programa tomará los parámetros definidos en el fichero de configuración "setupo_config.ini" para su funcionamiento. En este fichero puede establecer la interfaz de red en la que realizar la captura o el nombre del fichero PCAP, el nivel de los logs a mostrar, el

nombre de los ficheros de los módulos, parámetros de la base de datos, o la configuración para los avisos por correo electrónico. Consúltese el apéndice 1.

El hilo principal hace una lectura de los paquetes de la interfaz de red definida, o en su caso los obtiene del fichero PCAP. Por cada paquete capturado, realiza un análisis con la función "process_sniffed_packet()", en la que se evalúan todos los paquetes definidos en los módulos y sus parámetros. El paquete debe tener las capas definidas (ARP, IP, TCP, etc.) y los campos de cada una de ellas deben cumplir el parámetro definido (dirección destino, flags, tipo, etc.), y en tal caso se evalúa una expresión regular. En el caso de que el paquete cumpla las condiciones de uno, se genera un registro en la base de datos que principalmente contiene el identificador del tipo de paquete (un entero definido en el módulo) y el timestamp del paquete en cuestión. Véase el esquema de la figura 3 de este procedimiento.

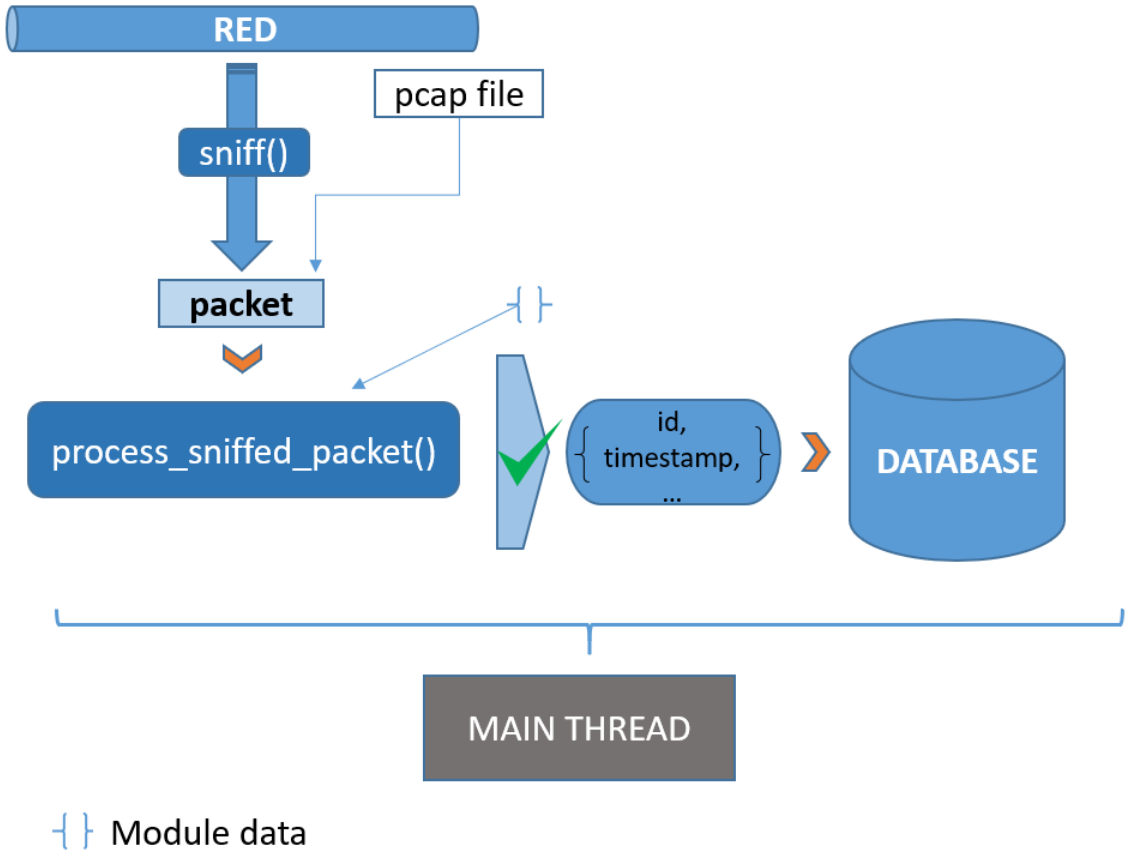


Figura 3. Esquema de funcionamiento del hilo principal en el core.

Un hilo secundario es el encargado de buscar patrones en los paquetes de red en base a los registros de la base de datos y mandar un aviso. Este segundo hilo ejecuta una función llamada "search_for_patterns()" de forma periódica según se configure. Esta función recorre todos los patrones en la base de datos, y evalúa cada uno en función del número y frecuencia de las repeticiones. Si se cumplen todas las condiciones de un mismo patrón llamará a otra función encargada de enviar un correo electrónico con el aviso. En el caso de que uno de los patrones ya haya sido notificado, comprobará que haya pasado un mínimo de tiempo definido antes de volver a notificar. El esquema de la figura 4 muestra este funcionamiento.

La función de aviso por una notificación es configurable y permite el envío de un correo electrónico con los detalles de la detección realizada.

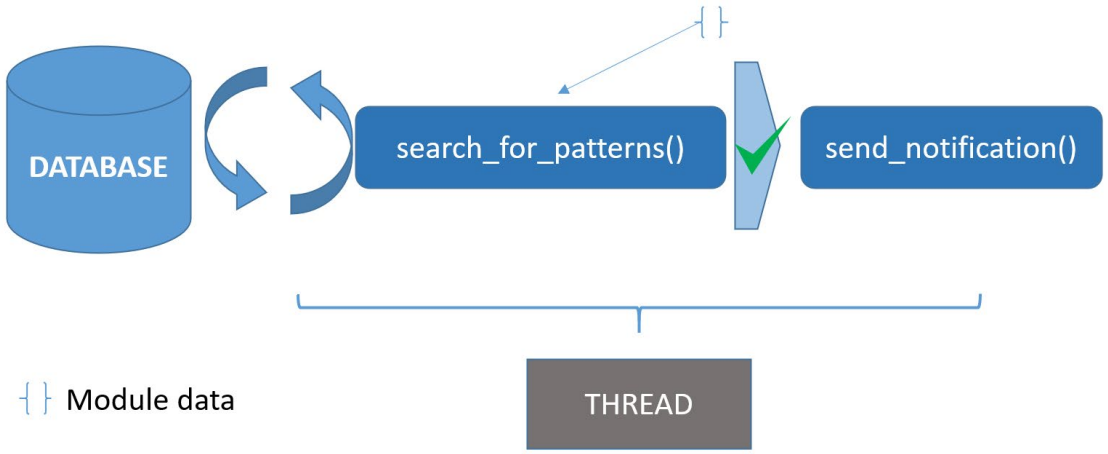


Figura 4. Esquema de funcionamiento de la búsqueda de patrones.

4.3 Descripción de módulo

Los módulos son ficheros de texto con formato JSON. Tienen definida una estructura fija con unos parámetros obligatorios en todos ellos que el programa toma para el análisis.

Cada módulo tiene 3 partes, siguiendo la figura 5. Un primer apartado con los parámetros generales, se incluyen campos como el nombre del módulo, versión, descripción o filtro del sniffer. Este último campo debe seguir un formato tipo BPF (Berkeley Packet Filter) o tipo tcpdump. Otro apartado con definiciones, que incluye los parámetros para caracterizar cada uno de los paquetes que pueden formar cierto patrón de comportamiento de malware. Además de nombre y descripción, permite definir una serie de protocolos y sus parámetros para esta caracterización, siempre que estén definidos en la librería Scapy, así como una expresión regular. Por ejemplo, para reconocer una petición "Who has …" del protocolo ARP, el parámetro "option" de la cabecera de este protocolo debe tener un valor de 1; o para una petición "Echo

reply" de ICMP (Ping), la capa ICMP debe tener el parámetro "type" con valor de 0. Un último apartado con patrones, donde se definen los parámetros y las distintas definiciones (paquetes) que forman cada patrón. Permite especificar el número de repeticiones, tiempo para estas repeticiones, identificadores de las definiciones y opciones para la notificación en caso de detectarse el patrón en cuestión. Se puede consultar el apéndice 2 con la descripción de cada parámetro para obtener más detalles.

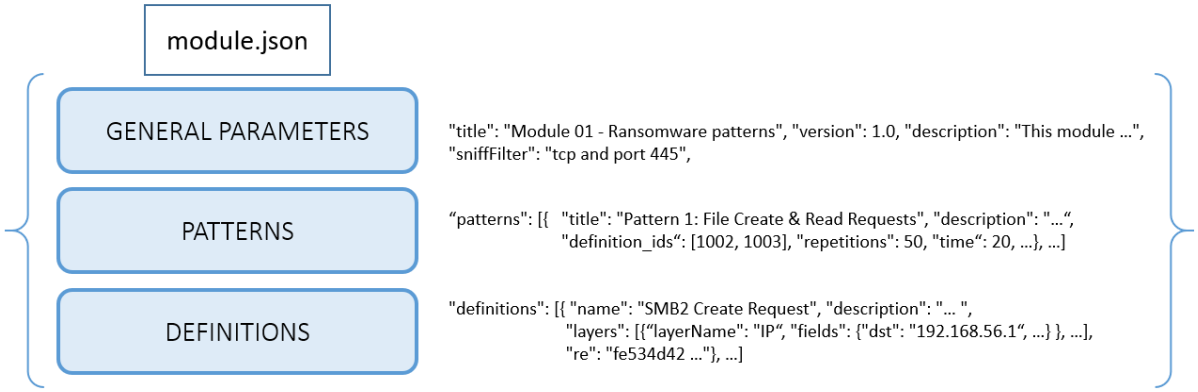


Figura 5. Esquema general básico de un módulo.

4.4 Desarrollo de módulos

Se han definido módulos para detección de ransomware y escaneos de máquinas y puertos en red.

El primer módulo se ha configurado para detectar actividad de cifrado de ficheros en red por parte de ransomware. Siendo Microsoft Windows el sistema operativo con un uso más extendido, emplea el protocolo Server Message Block (SMB) para compartir documentos y carpetas entre usuarios en multitud de entornos sobre todo en el ámbito ofimático. En el escenario inicialmente previsto donde probar la herramienta, se usaba la versión 2 de este protocolo el cual está implantado en máquinas desde Windows Vista hasta Windows 10, a pesar de que se está comenzando a usar la versión 3 en nuevas instalaciones para la que sería necesario tomar un enfoque distinto ya que el tráfico está cifrado.

SMB2 es un protocolo de nivel de aplicación que funciona sobre TCP en el puerto 139 o 445 [17]. Se pueden definir en el módulo dos capas y parámetros: la dirección destino del servidor para el nivel IP y el puerto destino 445 para TCP. El protocolo tiene una primera fase de establecimiento de la conexión y sesión con el servidor. Cuando se accede o se crea un fichero, se inicia la transacción con una petición tipo Create, que se caracteriza por tener el campo "Command" de la cabecera de SMB con valor 5 y además incluye el directorio y nombre del

fichero. En caso de que se vaya a hacer una operación de lectura, le seguirá la petición de lectura en la que el campo "Command" de la cabecera tendrá un valor de 8. Si es una de escritura, este campo tendrá un valor de 9 [16]. La frecuencia de acceso y escritura durante la actividad de cifrado del ransomware aumenta considerablemente, mandando esta serie de peticiones conforme se cifran los ficheros del equipo.

La cabecera de los paquetes tiene el identificador del protocolo con la cadena "0xfe534d42" que en ASCII corresponde a ".SMB", seguido de 8 bytes y posteriormente el campo "Command", que se puede definir en el parámetro de la expresión regular. Estos apartados del módulo para una petición Create quedan como:

```
"layers": [
 {
  "layerName": "IP",
  "fields": {
   "dst": "192.168.56.107"
  }
 },
 {
  "layerName": "TCP",
  "fields": {
   "dport": 445
  }
 }
],
"re": "fe534d42([0-9a-f][0-9a-f]){8}(?P<command>0500)"
```

Se definen también para las peticiones de lectura y escritura de ficheros. A la hora de configurar los patrones, se puede preparar uno de peticiones Create excesivas en un periodo de tiempo corto que indicaría un acceso masivo a ficheros. La aparición de numerosas peticiones inusuales de lectura y escritura que excedan el tráfico normal indicará un posible cifrado de archivos. Por ejemplo, para un escenario sencillo se puede configurar un número máximo de repeticiones de 50 en menos de 20 segundos para detectar este cifrado y evitar falsas alarmas. Estos valores se han obtenido realizando pruebas con capturas de tráfico normal y tráfico con cifrado de ficheros, y se han ido ajustando para evitar falsos positivos. No obstante, son unos valores sensibles al escenario configurado y se ven altamente afectados al cambiar de entorno de red. Si las peticiones de lectura y escritura tienen un identificador 1002 y 1003 respectivamente en el módulo, este último patrón quedaría como:

```
{
 "id": 1002,
 "title": "Pattern 2: excessive File Read & Write Requests",
 "definition_ids": [1002, 1003],
 "repetitions": 50,
 "time": 20,
 "timeUnit": "s",
 "repeat_notification": false,
 "min_repetition_time": 0
}
```


Los siguientes módulos se configuran para detectar escaneos de máquinas en la red y de puertos abiertos. En un escaneo activo de máquinas en una red local se mandan peticiones ARP del tipo “Who has …” al rango de direcciones IP a la espera de recibir respuesta de las activas. Se configura un paquete con una capa ARP con el parámetro “option” de la cabecera con un valor de 1. Otro método posible sería por medio de peticiones “Echo reply” del protocolo ICMP, si bien es menos utilizado por ser muy llamativo. Se configura una capa ICMP con un campo “type” de valor 0. Por otro lado, los escaneos de puertos a través del protocolo TCP que aprovechan las respuestas por defecto de los equipos para localizar los puertos abiertos y posteriormente aprovechar vulnerabilidades. En todos los paquetes definidos de este tipo de escaneos se establece la capa TCP y el campo “flags”. El flag de SYN se define con una “S”, SACK con “SA”, reset como “R” y los paquetes de un escaneo Xmas Tree (flags FIN, PSH y URG) como “FPU”. Este escaneo permite hacer un reconocimiento de puertos para saber si están abiertos o cerrados, se consideran puertos cerrados si los sistemas responden con reset. Este último caso queda configurado en el módulo de la siguiente forma:


```
{
 "id": 3004,
 "name": "3.4- DoS and Scanning patterns - TCP FPU",
 "description": "Xmas Port Scan - TCP FIN, PSH, URG",
 "layers": [
  {
   "layerName": "TCP",
   "fields": {
    "flags": "FPU"
   }
  }
 ],
 "re": ""
}
```

Capítulo 5. Pruebas y resultados

Se han realizado un conjunto de pruebas para probar las distintas funcionalidades. Se comprobará el funcionamiento del programa con el módulo para detectar cifrado por ransomware, así como los módulos de escaneos de máquinas y puertos en la red.

5.1 Cifrado de ficheros compartidos

Se prueba el funcionamiento del módulo para detectar el cifrado de ficheros compartidos simulando un escenario de un entorno ofimático que emplee el protocolo SMB. La herramienta de virtualización utilizada es Virtual Box en la versión 6.0.14 y las máquinas virtuales de una imagen de Windows 10 Enterprise ligera (7GB) con servicios mínimos [18]. Para configurar estas máquinas, se establece el adaptador de red de tipo Host only para tener una red aislada entre el ordenador y las máquinas virtuales en el rango 192.168.56.0/24. Se activa el uso compartido de archivos e impresoras en red y se comparte un directorio en la máquina servidor. Con un script "file_gen.py" se generan un número de ficheros de tamaño aleatorio entre ciertos valores proporcionados. El escenario resultante sigue el esquema de la figura 6 con tres máquinas en una red virtual: servidor de ficheros, cliente infectado que cifra los ficheros y el equipo sonda ejecutando el programa.

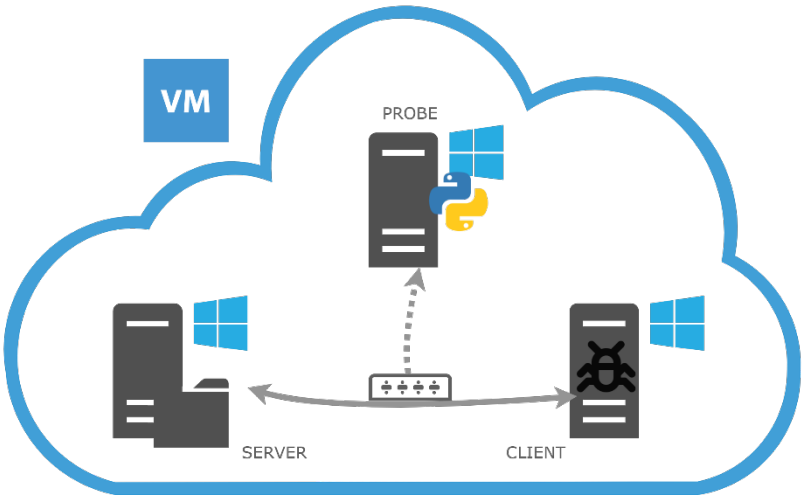


Figura 6. Escenario virtualizado de la prueba 1.

Las máquinas servidor y cliente tienen configurada 1 CPU y 1,5 GB de RAM, mientras que la sonda tiene 2 CPU y 2 GB de RAM. Se configura en el módulo la dirección destino del servidor 192.168.56.107 y el uso del puerto 445. Se ha hecho un script de cifrado de ficheros llamado "file_cipher.py" que encripta todos los archivos de los directorios indicados con AES256. Dado que se trata de un escenario sencillo y basado en la actividad de tráfico normal, los parámetros de número de repeticiones y tiempo para un patrón se establecen en 30

repeticiones en 20 segundos. Se realizan 3 test distintos variando parámetros como tamaño de los ficheros que se encriptan y número de scripts de cifrado en paralelo.

Previamente a iniciar el script de cifrado se ha hecho una aproximación para probar que no se obtienen falsos positivos. Se ha procedido a navegar entre los directorios compartidos, accediendo a 20 documentos y modificando 15 para simular un comportamiento de usuarios normales durante aproximadamente dos minutos. No obstante, este método no es concluyente y es necesario realizar pruebas con tráfico de un número de equipos más realista. Los resultados se muestran en la siguiente tabla 1 donde se indica el tiempo desde el inicio del cifrado hasta la detección y el número de ficheros que se cifran hasta la detección.

Tabla 1. Resultados de la prueba 1 de cifrado de ficheros

Test	Tamaño de ficheros	Nº scripts de cifrado	Falso positivo	Detección	Tiempo de detección	Ficheros encriptados
1	1-100 MB	1	No	Sí	25 s	28
2	1-100 MB	2	No	Sí	30 s	36
3	50-1000KB	2	No	Sí	58 s	100

Aunque en los tres casos se ha detectado el cifrado, los resultados son diferentes. El tamaño de los ficheros estaba entre 1 y 100 Megabytes en los dos primeros casos, aunque en el primero se activa un solo script de cifrado y en el segundo dos scripts en paralelo. En ambos casos se ha detectado en los primeros 25-30 segundos con tan solo 28 y 36 ficheros cifrados respectivamente. El funcionamiento ha sido correcto lanzando el aviso en un tiempo razonable.

En el tercer caso los ficheros son considerablemente de menor tamaño y se han vuelto a utilizar 2 scripts de cifrado, pero el tiempo de detección ha aumentado al doble y se han cifrado los 100 ficheros del directorio compartido. En este caso, a pesar de que se haya lanzado el aviso, se ha detectado un problema que puede llevar a un mal funcionamiento. Los ficheros son de menor tamaño y por lo tanto se cifran más por segundo, los paquetes se generan más rápido y el nivel de uso de CPU ha llegado al 100%. Aunque la máquina virtual solo tenía 2 CPU asignadas, si se escala el escenario a uno de alta velocidad real posiblemente aparezcan problemas de rendimiento. En este escenario la velocidad de encriptado no se ha visto limitada por la velocidad de la red al tratarse de un escenario virtual. Tras analizar los registros de la prueba, se observa que no se registran todas las peticiones generadas del protocolo SMB2 por lo que se produce pérdida de paquetes.

El carácter modular y estándar del programa ha llevado a realizar un procesamiento por paquetes a fin de detectar peticiones que formen un patrón de comportamiento de malware. Esto ocasiona que en escenarios de alta velocidad la máquina sonda se sature realizando el análisis paquete a paquete. Una posible solución al problema sería implementar el análisis por flujos que añadiría mayor complejidad.

5.2 Capturas de cifrado por ransomware

Con esta prueba se comprueba la funcionalidad del programa con capturas de tráfico en las que distintas variedades de ransomware cifran ficheros de un servidor de tipo NAS en un escenario similar al que se reflejaba en la figura 6 mostrada anteriormente. Un conjunto de equipos de usuarios conectados a un conmutador donde se conecta el servidor de ficheros compartidos y que redirige el tráfico por un puerto donde se conecta el equipo sonda. Se trata de un escenario virtual también y las máquinas ejecutan los sistemas operativos Windows 7 y Windows 10. Las capturas de tráfico se han obtenido del repositorio de ficheros PCAP para la evaluación de herramientas de detección de ransomware [19] [20].

Se ha evaluado el programa con ocho ejemplares de ransomware de este repositorio. Se han establecido los patrones del módulo de detección de ransomware con 100 repeticiones en 10 segundos, tras evaluar el tráfico del escenario y localizar el volumen de tráfico durante el cifrado que realiza el ransomware. Se ha comprobado que estos umbrales no hacen saltar falsos positivos con una aproximación de tráfico normal accediendo a 30 ficheros y modificando 20 de ellos durante dos minutos aproximadamente. Para cada prueba, se ha configurado el módulo con la dirección IP del servidor (192.168.1.5 o 10.0.2.5) y puerto utilizado por el protocolo SMB2 (139 o 445). Los resultados se muestran en la tabla 2 siguiente.

Tabla 2. Resultados de la prueba 2 con actividad de ransomware

Test	Ransomware	Detección	Tiempo de detección
1	Cerber 10-08-2016	Sí	8 s
2	Cerber 04-10-2016	Sí	15 s
3	Eris 02-08-2019	Sí	15 s
4	GandCrab 22-05-2019	Sí	6 s
5	Locky 27-04-2016	Sí	8 s
6	Phobos 08-05-2019	Sí	10 s
7	Ryuk 16-04-2019	Sí	11 s
8	TeslaCrypt 28-12-2015	Sí	5 s
9	WannaCry 16-05-2017	Sí	6 s

Se ha detectado la actividad de cifrado de ficheros en red para las nueve muestras de ransomware que se han probado independientemente de la variedad de este tipo de malware, puesto que el conjunto de programa y módulos se centra en descubrir actividad de cripto malware en general en la red y no de una variedad concreta ya conocida. Las muestras más antiguas datan del año 2015 mientras que las más recientes son del 2019.

5.3 Escaneo de clientes y puertos de red

Se prueba el funcionamiento del programa con los módulos para detectar escaneos activos de clientes en una red local y escaneo de los puertos abiertos de una máquina. El escenario configurado ha sido similar al anterior, pero en lugar de tener una máquina cliente que cifra archivos se ha configurado una máquina con Kali Linux en la versión 2020.01 como máquina atacante. El software utilizado para los escaneos ha sido Nmap al tratarse de la herramienta referente en este ámbito y la amplia variedad de posibilidades que ofrece. Los patrones de estos módulos se configuran con 50 repeticiones en 10 segundos para lanzar un aviso, tras evaluar el volumen de este tipo de peticiones bajo condiciones normales del escenario.

Se realiza un primer test para detectar un escaneo de máquinas por peticiones tipo "Who has …" de ARP con el comando "nmap -sP 192.168.56.0/24" que realiza un sondeo ping sin puertos de clientes en la red [20], y en los casos siguientes se efectúan distintos tipos de escaneos de puertos sobre la máquina en la dirección IP 192.168.56.110 cuyos resultados se pueden ver en la siguiente tabla 3.

Tabla 3. Resultados de la prueba 3 de escaneos.

Test	Tipo escaneo	Comando nmap	Falso positivo	Detección	Tiempo de detección
1	Hosts	nmap -sP	No	Sí	9 s
2	TCP SYN	nmap -sS	No	Sí	10 s
3	TCP ACK	nmap -PA	No	Sí	8 s
4	Xmas	nmap -sX	No	Sí	10 s
5	SYN Stealth	nmap -sS	No	Sí	11 s

Se ha seguido el mismo procedimiento para evaluar los distintos tipos. Primero se activa el programa en la máquina sonda y desde la máquina atacante se inicia el escaneo, en este caso del tipo Xmas Tree, como se ve en la figura 7 siguiente.

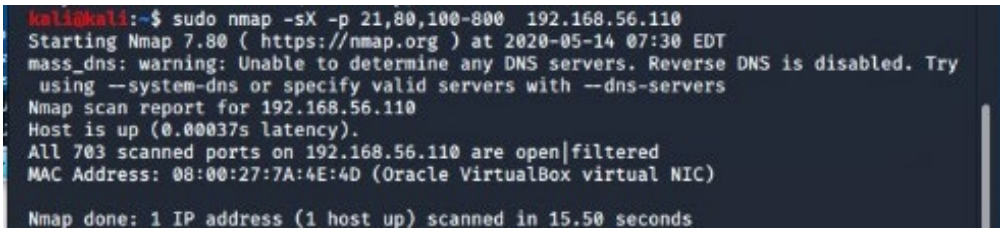
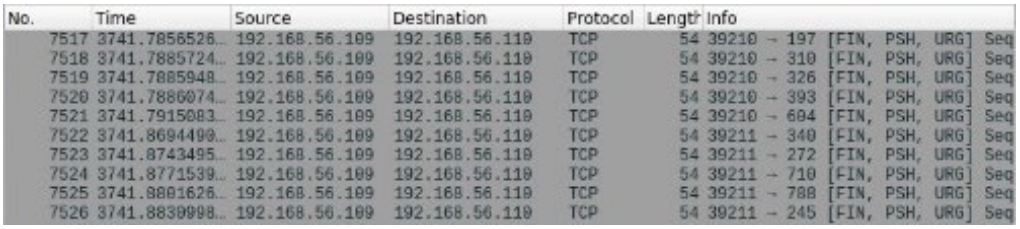


Figura 7. Escaneo Xmas con Nmap

Con este escaneo se generan una serie de paquetes TCP hacia la máquina en la IP 192.168.56.110 que se puede ver en la figura 8 a continuación, una captura de paquetes de Wireshark realizada durante el escaneo.



No.	Time	Source	Destination	Protocol	Length	Info
7517	3741.7856526…	192.168.56.109	192.168.56.110	TCP	54	39210 → 197 [FIN, PSH, URG] Seq
7518	3741.7885724…	192.168.56.109	192.168.56.110	TCP	54	39210 → 310 [FIN, PSH, URG] Seq
7519	3741.7885948…	192.168.56.109	192.168.56.110	TCP	54	39210 → 326 [FIN, PSH, URG] Seq
7520	3741.7886074…	192.168.56.109	192.168.56.110	TCP	54	39210 → 393 [FIN, PSH, URG] Seq
7521	3741.7915083…	192.168.56.109	192.168.56.110	TCP	54	39210 → 604 [FIN, PSH, URG] Seq
7522	3741.8694490…	192.168.56.109	192.168.56.110	TCP	54	39211 → 340 [FIN, PSH, URG] Seq
7523	3741.8743495…	192.168.56.109	192.168.56.110	TCP	54	39211 → 272 [FIN, PSH, URG] Seq
7524	3741.8771539…	192.168.56.109	192.168.56.110	TCP	54	39211 → 710 [FIN, PSH, URG] Seq
7525	3741.8801626…	192.168.56.109	192.168.56.110	TCP	54	39211 → 788 [FIN, PSH, URG] Seq
7526	3741.8830998…	192.168.56.109	192.168.56.110	TCP	54	39211 → 245 [FIN, PSH, URG] Seq

Figura 8. Paquetes en Wireshark durante escaneo Xmas.

Unos segundos después de iniciar el proceso, salta el aviso en el programa indicando que se han detectado dos patrones de escaneos: un escaneo de máquinas por peticiones ARP y un escaneo de puertos de tipo Xmas que se puede ver en la figura 9.

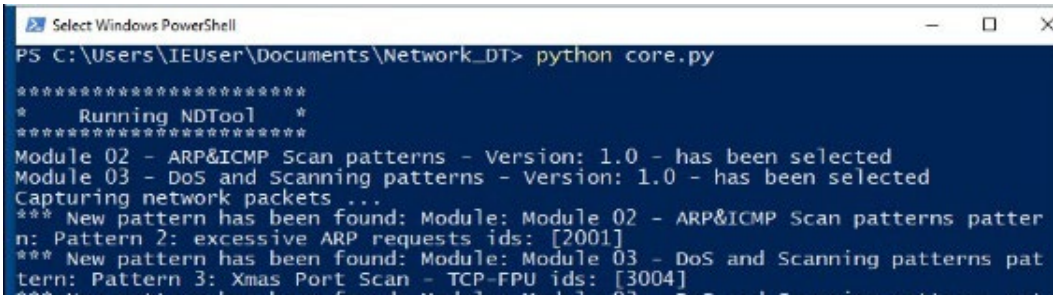


Figura 9. Aviso en consola de patrón de escaneo ARP y Xmas.

Tanto los escaneos por ARP como los de TCP con los flags SYN o ACK, Xmas con los flags FIN, PSH y URG han sido detectados en torno a los 10 segundos desde el inicio del escaneo. El programa ha lanzado el aviso para los cinco tipos de escaneo que se han probado en un tiempo muy razonable de cara a detectar una intrusión que emplee este tipo de escaneo para realizar movimientos laterales en la red.

5.4 Análisis y discusión

Los parámetros empleados de número de repeticiones y tiempo de estas repeticiones son muy sensibles al escenario en el que analiza el tráfico. Esto implica que se deben adaptar a las características del marco y variará según el número de máquinas y actividad que realicen de normal. En una situación con pocas máquinas resulta de mayor facilidad detectar un incremento de tráfico desde una de ellas que en uno más numeroso donde puede quedar camuflado al existir mayor interacción entre ellas. Una posible mejora es realizar el control de peticiones por cliente en lugar de a nivel global, de forma que estos parámetros quedan más estables y fluctuarían menos entre escenarios. Solo sería necesario un análisis previo del comportamiento

de los usuarios a fin de caracterizar el volumen de tráfico en condiciones normales sin infecciones.

El método elegido para detectar los patrones de tráfico por número de operaciones puede generar falsos positivos en el caso de que los usuarios realicen operaciones de gran volumen como copiar o mover carpetas entre directorios si no se ha previsto este tipo de operaciones a la hora de establecer los umbrales. Igualmente, en los resultados anteriores no se ha tenido en cuenta el tiempo en recibir la notificación por correo electrónico, ya que dependerá de la congestión de los servidores de correo y otros parámetros en función del servicio que se utilice, ya sean correo interno o externo.

Hay que mencionar, además, se ha detectado un posible problema de rendimiento en escenarios de alta velocidad en la prueba 1, debido al análisis paquete por paquete realizado por el carácter modular que requiere una amplia compatibilidad para detectar gran variedad de ataques. Si bien es cierto que la máquina sonda no tenía configurados los recursos de una máquina completa con varios núcleos de CPU y más RAM, no sería escalable para este tipo de escenarios por el nivel alto de recursos que serían necesarios. Una posible mejora es realizar el análisis de tráfico por flujos para evitar saturación por el alto número de paquetes.

Así mismo, cabe señalar que quedaría por probar la herramienta en un escenario más realista en número de máquinas y tráfico generado a fin de comprobar que no se obtienen falsos positivos. El escenario de prueba elaborado no refleja una situación real en cuanto a estos parámetros mencionados y no ofrece resultados concluyentes de esta medida. Sería necesario realizar pruebas con un número de equipos acorde al escenario de uso y así ajustar parámetros para una valoración completa.

Capítulo 6. Conclusiones

Se ha desarrollado un programa de tipo modular que analiza el tráfico de la red y si encuentra indicios de actividad sospechosa, lanza un aviso para los administradores IT. Se divide principalmente en dos partes: el programa en sí y los módulos. El programa realiza las funciones comunes necesarias para analizar la actividad de red, como leer las trazas y su contenido, generar una serie de registros, y dar posibles avisos en caso de encontrar patrones de actividad de malware sin requerir intervención del usuario. La actividad por detectar queda definida en los módulos y se basa en ciertos paquetes y peticiones característicos de los ataques de red, así como en patrones y su frecuencia de repetición. Se han definido módulos para detección de cifrado de archivos compartidos por ransomware y escaneos de máquinas y puertos en red.

Se ha comprobado su funcionamiento llevando a cabo un conjunto de pruebas para probar las distintas funcionalidades. Se han formado escenarios virtuales para detectar la actividad de cifrado por ransomware o escaneos de máquinas y puertos en la red, así como una evaluación con distintas capturas de tráfico de actividad de cifrado de distintas variedades de ransomware. En todas las pruebas se han lanzado los avisos correspondientes tras la correcta detección. Se ha detectado un posible problema de rendimiento para un escenario de redes de alta velocidad por el uso de la base de datos que actúa como cuello de botella.

Bibliografía

[1] EUROPOL, «Internet Organised Crime Threat Assessment (IOCTA) 2018,» Europol - European Police Office, 2019. [En línea]. Available: https://op.europa.eu/s/n3PC. [Último acceso: 24 Abril 2020].

[2] Malwarebytes Labs, «2020 State of Malware Report,» Febrero 2020. [En línea]. Available: https://resources.malwarebytes.com/files/2020/02/2020_State-of-Malware-Report.pdf. [Último acceso: 24 Abril 2020].

[3] EUROPOL, «Internet Organised Crime Threat Assesment (IOCTA) 2019,» 2019. [En línea]. Available: https://europa.eu/!Xc83qG. [Último acceso: 20 Abril 2020].

[4] L. H. Newman, «The Biggest Cybersecurity Crises of 2019 So Far,» WIRED, 7 Mayo 2019. [En línea]. Available: https://www.wired.com/story/biggest-cybersecurity-crises-2019-so-far/. [Último acceso: 26 Abril 2020].

[5] BBC, «Cyber-attack: Europol says it was unprecedented in scale,» BBC, 13 Mayo 2017. [En línea]. Available: https://www.bbc.com/news/world-europe-39907965. [Último acceso: 24 Abril 2020].

[6] P. C. C. H. U. Bayer, «Scalable, Behavoir-Based Malware Clustering,» *Network and Distributed System Security,* 2009.

[7] OPSWAT, «Understanding Heuristic-based Scanning vs. Sandboxing,» 2015. [En línea]. Available: https://www.opswat.com/blog/understanding-heuristic-based-scanning-vs-sandboxing. [Último acceso: 11 Mayo 2020].

[8] J. Cloonan, «Advanced Malware Detection - Signatures vs. Behavoir Analysis,» Info Security, 2017. [En línea]. Available: https://www.infosecurity-magazine.com/opinions/malware-detection-signatures/. [Último acceso: 10 05 2020].

[9] «SDI-Tipos de IDS,» Administración de Sistemas Operativos, [En línea]. Available: http://www.adminso.es/index.php/SDI-I-TIPOS_DE_IDS. [Último acceso: 11 Mayo 2020].

[10] «What is an Intrusion Prevention System?,» Palo Alto Networks, [En línea]. Available: https://www.paloaltonetworks.com/cyberpedia/what-is-an-intrusion-prevention-system-ips. [Último acceso: 11 Mayo 2020].

[11] «Snort FAQ/Wiki,» Snort, [En línea]. Available: https://www.snort.org/faq. [Último acceso: 11 Mayo 2020].

[12] Python Software Foundation, «Python,» [En línea]. Available: https://www.python.org/. [Último acceso: 10 Abril 2020].

[13] «Scapy Docs,» [En línea]. Available: https://scapy.readthedocs.io/. [Último acceso: 10 Marzo 2020].

[14] «Regex,» [En línea]. Available: https://pypi.org/project/regex/. [Último acceso: 2 Marzo 2020].

[15] «SQLAlchemy,» [En línea]. Available: https://www.sqlalchemy.org/. [Último acceso: 20 Marzo 2020].

[16] Mark Loman, SophosLabs, «Ransomware behavior report - How Ransomware Attacks,» Noviembre 2019. [En línea]. Available: https://www.sophos.com/en-us/medialibrary/PDFs/technical-papers/sophoslabs-ransomware-behavior-report.pdf. [Último acceso: 29 Abril 2020].

[17] Microsoft Corporation, «[MS-SMB2]: Server Message Block (SMB) Protocol Versions 2 and 3,» Microsoft, 2019. [En línea]. Available: https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-smb2/5606ad47-5ee0-437a-817e-70c366052962?redirectedfrom=MSDN.

[18] Microsoft, «Windows 10 Virtual Machine,» [En línea]. Available: https://developer.microsoft.com/en-us/microsoft-edge/tools/vms/.

[19] D. M. ,. E. M. M. I. Eduardo Berrueta, «Open Repository for the Evaluation of Ransomware Detection Tools,» IEEE Dataport, 2020. [En línea]. Available: http://dx.doi.org/10.21227/qnyn-q136. [Último acceso: 15 Mayo 2020].

[20] D. M. E. M. y. M. I. E. Berrueta, «Open Repository for the Evaluation of Ransomware Detection Tools,» *IEEE Access,* vol. 8, nº 10.1109/ACCESS.2020.2984187, pp. 65658-65669, 2020.

[21] G. Lyon, «Guía de referencia de Nmap,» NMAP.ORG, [En línea]. Available: https://nmap.org/man/es/index.html. [Último acceso: 4 Mayo 2020].

[22] Sophos, «Ransomware: The Cyberthreat that Just Won't Die,» Diciembre 2019. [En línea]. Available: https://secure2.sophos.com/en-us/security-news-trends/whitepapers/gated-wp/ransomware-cyberthreat.aspx?cmp=26062#GetWhitepaper. [Último acceso: 30 Abril 2020].

Apéndices

1. Setup Configuration File Documentation (setup_config.ini)

[key: example value *(Notes)Value type. description***]**

Network Detection Tool Configuration Setup File

[Main]

project_dir = C:/Users/cabal/Software/PyCharm/PycharmProjects/Network_DT *Project directory path*

log_file = network_DT.log *Logging file name*

File mode: w - overwrite, a - append

log_file_mode = w *Logging file mode, overwrite (w) or append (a)*

CRITICAL = 50, ERROR = 40, WARNING = 30, INFO = 20, DEBUG = 10

log_level = 20 *Logging level selection, show all logs above this level. Ex. 20 – it will show INFO, WARNING, ERROR and CRITICAL logs only, dismissing DEBUG ones*

[Sniffer]

iface = Ethernet 2 *Network Interface to sniff*

For live sniffing, just comment the line

cap_file = Captura_cipher_10.pcapng *PCAP file to evaluate. Comment the line to sniff the network*

[Search_patterns]

search_time = 10 *Seconds. How often to perform database pattern searches*

min_repetition_time = 60 *Seconds. Minimum time between notifications for the same pattern. 0 to notify all*

[DB]

db_file = sqlite:///sqlalchemy_network_DT.db *Database filename. It is generated after running the database generator sqlalchemy_NDT_declarative.py*

db_reset = True *Boolean. Delete all entries in DB at start*

db_delete_old_timer = 86400 *Integer. (seconds) Timer value to perform a database old records deletion.*

db_delete_old_expire_time = 86400 *Integer. (seconds) Database records expiring time.*

[Modules]

modules_dir= ${Main:project_dir}/ *Module files directory path*

modules = ["module01_ransomware.json", "module02_scans.json"] *Strings Array. Modules filenames*

[Email]

send_email = True *Boolean. Notify pattern found by email*

#Server parameters # For SSL

port = 465 *Sender server email port*

smtp_server = smtp.gmail.com *Sender server*

#Email sender parameters # Sender email address

sender_email = networkhunter.notif.service@gmail.com *Sender email address*

password = testNHpython *Sender email password*

receiver_email = vahen43173@mailop7.com *Receiver email address*

Mail Message Receiver parameters [display_name] <username@domain>

display_name = Manager *Email receiver displaying name*

username = manager *Email receiver displaying username*

domain = domain *Email receiver displaying domain*

[DEFAULT]

All parameters in this section are the default values to take if they have not been defined in sections above.

2. Module Standard Schema Documentation

Modules must follow the standardization of this json schema.

[key: example value *(Notes)Value type. description***]**

{

"id": 1, *Integer. Module identifier.*

"title": "Module 01 - Ransomware patterns", *String. Module title*

"version": "1.0", *String. Module version number*

"description": "This module contains patterns found in file shares network traffic when encrypting files ", *String. Module description*

"sniffFilter": "tcp and port 445", *String. Sniffer BPF/TCPdump filter.*

"patterns": [*Array. It contains the different patterns to search in the DB*

{

"id": 1001, *Integer. Pattern identifier. ABBB (A: Module number, B: pattern number)*

"title": "Pattern 1: excessive File Create & Read Requests", *String. Pattern title.*

"definition_ids": **[**1001, 1002**]**, *Array. Pattern title.*

"repetitions": 10**,** *Integer. Pattern repetitions in the DB*

"time": 10**,** *Integer. Pattern repetition max time*

"timeUnit": "s", *String.* s / m / h *Time unit. Seconds as default*

"repeat_notification": true, *Boolean. Repeat notifications of this pattern*

"min_repetition_time": 60 *Integer. Seconds. Minimum time before notifying the same pattern again. timestamp format. 0 to notify all*

}

],

"definitions": [*Array. It contains the definitions, network packets to save records on the DB*

{

```
        "id": 1001,  Integer. Definition identifier. ABBB (A: Module number, B: definition number)
        "name": "1.1- Ransomware pattern - SMB2 Create Request",  String. Definition title.
        "description": "SMB2 Create Request ... ",  String. Definition description.
        "layers": [  Array. Layer parameters to check on for each packet
                {
                 "layerName": "IP",  String. Layer name
                 "fields": {
                  "src": null,
                  "dst": "10.160.1.1"  String. Layer field and value
                  }
                }
            ]
      ],
      "re": "fe534d42([0-9a-f][0-9a-f]){8}(?P<command>0500)"  String. Regular expression
    }
  ]
}
```

3. NetworkDT documentation

NetworkDT is a detection tool that works as a packet sniffer. It works in combination with the json modules where malware patterns are defined. This tool creates records on the database for each sniffed packet that is also defined on the modules. The tool has been tested with Python 3.8.

Python 3.8 is needed to run this tool as well as Ncap or Windump. There are some Python libraries that the program requires:

- Scapy: "Scapy is a powerful Python-based interactive packet manipulation program and library".
  Installation: https://pypi.org/project/scapy/
  Documentation: https://scapy.readthedocs.io/en/latest/
- Regex: "This regex implementation is backwards-compatible with the standard 're' module, but offers additional functionality".
  Installation: https://pypi.org/project/regex/
  Documentation: https://docs.python.org/3/library/re.html
- SQLAlchemy: "SQLAlchemy is the Python SQL toolkit and Object Relational Mapper that gives application developers the full power and flexibility of SQL. SQLAlchemy provides a full suite of well known enterprise-level persistence patterns, designed for efficient and high-performing database access, adapted into a simple and Pythonic domain language".
  Installation: https://pypi.org/project/SQLAlchemy/
  Documentation: https://www.sqlalchemy.org/
  All these packages can be easily installed through the pip installer.

The file cipher script (file_cipher.py) needs cryptography library: "cryptography includes both high level recipes and low level interfaces to common cryptographic algorithms such as symmetric ciphers, message digests, and key derivation functions".

Installation: https://pypi.org/project/cryptography/
Documentation: https://cryptography.io/en/latest/

Before executing the program, a couple steps need to be followed. First, database must be created with "sqlalchemy_NDT_declarative.py" script, a file named "sqlalchemy_network_DT.db" should have been created. Program logs will be listed in the "network_DT.log" file.

Many program parameters can be set up in the configuration file "setup_config.ini" like the sniffing interface or capture file, mail notification server parameters, database configuration, etc. Then, just run "core.py".